1. 实例： 访问图形编辑器中的对象

可以使用 VBS WinCC 对所有图形编辑器对象进行访问，以使图形运行环境动态化。 根据变量或周期性（例如闪烁）情况，可在执行操作（例如在按钮上单击鼠标）时使图形对象动态化。

以下示例说明如何在鼠标单击后更改图形对象。

步骤

在以下示例中，每次单击鼠标时运行系统中圆的半径都会设置为 20：

```
Dim objCircle

Set objCircle= ScreenItems("Circle1")

objCircle.Radius = 20
```

2. 实例： 定义对象的颜

图形对象的颜色通过 RGB 值（红/绿/蓝）定义。 可以设置或读出图形对象的颜色值。

步骤

以下示例将“ScreenWindow1”的填充颜色定义为蓝色：

```
Dim objScreen

Set objScreen = HMIRuntime.Screens("ScreenWindow1")

objScreen.FillStyle = 131075

objScreen.FillColor = RGB(0, 0, 255)
```

3. 例：如何组态语言切换

可使用 VBS 切换 WinCC 的运行系统语言。 最常用的是包含相应语言代码的按钮，这些按钮位于项目的起始页上。

在 VBS 中通过使用国家代码（例如，1031 表示德语 - 默认，1033 表示英语 - 美国等）指定运行系统语言。有关所有国家代码的汇总，请参见标题为“区域方案 ID (LCID) 图”的主题下的 VBScript 基本知识。

步骤:

使用按钮上的“Mouse click”事件创建 VBS 动作，输入以下动作代码将运行系统语言切换为德语：

```
HMIRuntime.Language = 1031
```

4. 实例： 禁用运行系统

简介

可以使用 VBS 终止 WinCC 运行系统，例如，通过鼠标单击，依靠变量值或其它事件（例如，启动运行系统时密码的多次错误输入）。

要执行的操作

以下示例会终止 WinCC 运行系统：

```
HMIRuntime.Stop
```

5. 实例： 全局组态画面更改

简介

VBS 可用于启动全局画面更改，因而会在分布式系统的客户机上显示服务器中的画面。 为此，服务器的服务器前缀必须位于目标画面之前。

要执行的操作

为按钮组态以下画面更改代码，例如：

```
HMIRuntime.BaseScreenName = "Serverprefix::New screen"
```

6. 实例： 通过属性组态画面更改

简介

如果在组态中使用分区画面（例如，在用户界面的基本画面标题和操作栏中和用于实际画面显示的嵌入画面窗口中），应使用画面窗口的属性组态画面更改。

为了显示其它画面，必须更改“ScreenName”画面窗口的属性。 必须在同一画面中对动作和画面窗口进行组态。

要执行的操作

在以下示例中，执行动作时“ScreenWindow”画面窗口中会显示“test.pdl”画面：

```
Dim objScrWindow

Set objScrWindow = ScreenItems("ScreenWindow")

objScrWindow.ScreenName = "test"
```

7. 实例： 通过 Trace 组态诊断输出

简介

如果已将 GSC 诊断窗口插入画面中，则可以使用 Trace 命令在运行系统的诊断窗口中显示诊断输出。

GSC 诊断按调用的先后顺序发出包含在动作中的 Trace 方法。这也适用于在动作中调用的过程中的 Trace 指令。Trace 指令的有目的执行（例如针对变量值的输出）可实现对动作进度以及在动作中调用的过程的跟踪。Trace 指令以“HMIRuntime.Trace(<Ausgabe>)”形式输入。

GSC 诊断显示来自 C 和 VBS 的跟踪输出。

要执行的操作

以下示例将文本写入诊断窗口中：

```
HMIRuntime.Trace "Customized error message"
```

8. 实例： 写入变量值

可以用 VBS 将变量值写入 PLC 中，例如通过在按钮上单击鼠标来指定设定值，或设置内部变量值，以触发其它动作。

下面涉及和介绍了多种写入变型。

1) **简单写入**

在以下示例中，将值写入“Tag1”变量内：

```
HMIRuntime.Tags("Tag1").Write 6
```

这是最简单的写入形式，因为不会生成任何对象引用。

2) **通过对象引用写入**

在以下示例中，将创建变量对象的本地副本并将值写入“Tag1”内：

```
Dim objTag

Set objTag = HMIRuntime.Tags("Tag1")

objTag.Write 7
```

3) 通过利用引用，可以在写入之前使用变量对象。 可以读取变量值，进行计算，并再次写入：

```
Dim objTag

Set objTag = HMIRuntime.Tags("Tag1")

objTag.Read

objTag.Value = objTag.Value + 1

objTag.Write
```

4) **同时写入**

通常，待写入的值会传送到变量管理，然后重新开始对动作进行处理。 但某些情况下，必须确保实际写入了值之后才能重新开始对动作进行处理。

此类写入通过将附加的可选参数指定为值 1 来实现：

```
Dim objTag

Set objTag = HMIRuntime.Tags("Tag1")

objTag.Value = 8

objTag.Write ,1
```

说明

请注意，这种调用比标准调用使用的时间要长。 除此之外，持续时间还取决于通道和 AS。

这类写入遵从 C 脚本中的 SetTagXXXWait() 调用

5） 通过状态处理写入

为了确保成功写入值，必须在写入过程之后执行错误检查或确定变量状态。

为此，执行写入操作后需检查“LastError”属性。 测试成功（即成功放置任务）后，即检查变量状态。

对于写入任务，过程的当前状态尚不确定。 要确定该状态，必须读取变量。 读取过程之后“质量代码”属性中指定的值会提供变量状态指示，如有必要，还会涉及发生故障的 AS 连接。

在以下示例中，将写入“Tag1”变量。 如果写入期间出现错误，全局脚本诊断窗口中会显示错误值和错误描述。 最后，检查质量代码。 如果质量代码不是 OK (0x80)，便在诊断窗口中显示该代码

```
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 9
If 0 <> objTag.LastError Then
HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " & objTag.ErrorDescription &
vbCrLf
Else
objTag.Read
If &H80 <> objTag.QualityCode Then
HMIRuntime.Trace "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf
End If
End If
```

说明

写入变量后，由于不知道哪一质量代码在过程中用于管理变量，因此局部变量对象的 QualityCode 属性会设置为“BAD 已不能用”。

9. 实例： 如何读取变量值

简介

可以用 VBS 读取变量值并对其执行进一步的处理。 这样便可以执行诸如通过在按钮上单击鼠标来获取系统状态信息或执行计算的操作。

下面涉及和介绍了多种读取变型。

1） 简单读取

在以下示例中，将读取“Tag1”的值并在全局脚本诊断窗口中显示该值：

```
HMIRuntime.Trace "Value: " & HMIRuntime.Tags("Tag1").Read & vbCrLf
```

这是最简单的读取形式，因为不会生成任何对象引用。

2） 通过对象引用读取

在以下示例中，将生成变量对象的本地副本，读取该变量值并在全局脚本诊断窗口中显示该值：

```
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
HMIRuntime.Trace "Value: " & objTag.Read & vbCrLf
```

通过利用引用可以使用变量对象。 可以读取变量值，进行计算，并再次写入：

```
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
```

使用 Read 方法将已读取的过程变量添加到图像，从该刻起这些变量会通过 AS 周期性请求。 如果该变量已存在于图像中，则会返回其中包含的值。

对于关闭画面，变量动作会再次结束。

说明

如果全局脚本动作中请求变量，则在进入 WinCC 运行系统的整个过程中，该变量保持已注册状态

3） 直接读取

通常，变量值从变量图像读取。 但在某些情况下，例如为了同步快速过程，可能需要直接从 AS 读取变量值。

如果将读取过程的可选参数设置为 1，则不会周期性地登录变量，而是通过 AS 单次请求该值。

```
Dim objTag

Set objTag = HMIRuntime.Tags("Tag1")

HMIRuntime.Trace "Value: " & objTag.Read(1) & vbCrLf
```

说明

请注意，这种调用比标准调用使用的时间要长。 除此之外，持续时间还取决于通道和 AS。

在执行周期性 C 动作的情况下，必须避免该类调用，因为这是引起性能问题的主要原因。

该类读取过程相当于 C 脚本中的 GetTagXXXWait() 调用。

4) 通过状态处理读取

为了确保值有效，应在读取之后进行检查。 这通过控制质量代码来执行。

在以下示例中，将读取“myWord”变量，然后检查 QualityCode。 如果质量代码未对应 OK (0x80)，则在全局脚本诊断窗口中显示 LastError、ErrorDescription 和 QualityCode 属性。

```
Dim objTag

Set objTag = HMIRuntime.Tags("Tag1")

objTag.Read

If &H80 <> objTag.QualityCode Then

HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " & objTag.ErrorDescription & vbCrLf & "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf

Else

HMIRuntime.Trace "Value: " & objTag.Value & vbCrLf

End If
```

说明

如果读取期间出现错误，则 QualityCode 会设置为 BAD NON-SPECIFIC。 因此，只需在读取之后检查

1） 示例： 写入对象属性

VBS 可实现对所有图形编辑器画面对象的属性的访问。运行期间可以读出各个属性以便进行修改或更改。

以下示例说明了各种访问形式。

1） 属性的简单设置

在以下示例中，画面中包含的“Rectangle1”对象的背景颜色被设置为红色

```
ScreenItems("Rectangle1").BackColor = RGB(255,0,0)
```

这是最简单的写入形式，因为不会生成任何对象引用

说明

如果不通过对象引用完成操作，则智能感知中只提供标准属性。

本示例中使用的表达式形式仅适用于图形编辑器。 对于全局脚本中的模拟动作，应使用 HMIRuntime 对象访问相应对象

2） 通过对象引用设置属性

在以下示例中，将创建对画面中所包含“Rectangle1”对象的引用，并使用 VBS 标准函数 RGB() 将背景设置为红色：

```
Dim objRectangle

Set objRectangle = ScreenItems("Rectangle1")

objRectangle.BackColor = RGB(255,0,0)
```

必须更改多个对象属性时，引用非常有用。 使用智能感知时，该过程即会列出所有对象属性。

说明

本示例中使用的表达式形式仅适用于图形编辑器。 对于全局脚本中的模拟动作，应使用 HMIRuntime 对象访问相应对象

3） 通过画面窗口设置属性

图形编辑器中的 VBS 提供两种可行的画面超越访问方法：

- 使用“ScreenItems”通过画面窗口的 Screen 对象
- 使用“HMIRuntime.Screens”通过基本画面

通过画面窗口引用

以下示例中，在从属画面窗口中更改矩形的颜色。 相应脚本在画面窗口“ScreenWindow1”所处的画面“BaseScreen”中执行。 此画面窗口会显示包含名称为“Rectangle1”的“Rectangle”类型对象的画面。

```
Sub OnLButtonUp(ByVal Item, ByVal Flags, ByVal x, ByVal y)

Dim objRectangle

Set objRectangle =
ScreenItems("ScreenWindow1").Screen.ScreenItems("Rectangle1")

objRectangle.BackColor = RGB(255,0,0)

End Sub
```

4) 通过基本画面引用

可通过 HMIRuntime.Screens 引用具有待修改对象的画面。该画面相对于基本画面的规范通过以下访问代码进行定义：

```
[<Grundbildname>.]<Bildfenstername>[:<Bildname>]... .<Bildfenstername>[:<Bildname>]
```

在以下示例中，将创建对“Rectangle1”画面中包含的“Screen2”对象的引用，并将背景颜色设置为红色。

这种情况下，画面“Screen2”位于“Screen1”中。 “Screen1”显示在基本画面“BaseScreen”中。

```
Dim objRectangle

Set objRectangle =
HMIRuntime.Screens("BaseScreen.ScreenWindow1:Screen1.ScreenWindow1:Screen2")
.ScreenItems("Rectangle1")

objRectangle.BackColor = RGB(255,0,0)
```

无需指定画面名称。可以通过画面窗口名称唯一地访问某一画面。因此，只需指定画面窗口的名称，如下示例所示：

```
Dim objRectangle

Set objRectangle =
HMIRuntime.Screens("ScreenWindow1.ScreenWindow2").ScreenItems("Rectangle1")

objRectangle.BackColor = RGB(255,0,0)
```

这种访问类型可实现在不同画面中访问画面窗口中的对象。 就画面模块技术而言，这是特别有趣的一点。

5） 利用返回值使属性动态化

基于属性的动作不仅能由事件触发或周期性触发，而且能直接通过动作使属性动态化。

在以下示例中，通过返回值使对象的背景颜色动态化。 例如，传送的值可能来自 PLC 中事件的评估，并用于运行状态的图形显示：

```
Function BackColor_Trigger(ByVal Item)

BackColor_Trigger = RGB(125,0,0)

End Function
```

说明

如果通过脚本的返回值使具有 VBS 动作的对象属性动态化，则只有在相对于上次运行的脚本对象属性值发生更改时才会写入该值。 如果该值已在另一位置发生更改则无效。

因此，通过从另一位置（例如，其它 C 脚本或 VBS 脚本）的返回值来更改由 VBS 动作生成的动态属性是非法的。

如果不遵守这一点，则结果可能是错误的值。

2） 实例： 通过 VBS 组态数据库连接

简介

以下示例说明如何通过 ODBC 驱动器组态 Access 数据库链接。

- 示例 1 将 WinCC 的变量值写入 Access 数据库中。
- 示例 2 从数据库读取值并将其写入 WinCC 变量中。

这些示例不包含任何处理故障。

1.通过 WINCC_DATA 表和 ID 在其中作为自动值的的列（ID,TagValue）来创建 Access 数据库。

2.设置名称为“SampleDSN”的 ODBC 数据源，引用以上 Access 数据库。

3.编程。

3） 示例 1

```
Dim objConnection

Dim strConnectionString

Dim lngValue

Dim strSQL
```

```
Dim objCommand

strConnectionString = "Provider=MSDASQL;DSN=SampleDSN;UID=;PWD=;"

lngValue = HMIRuntime.Tags("Tag1").Read

strSQL = "INSERT INTO WINCC_DATA (TagValue) VALUES (" & lngValue & ");"

Set objConnection = CreateObject("ADODB.Connection")

objConnection.ConnectionString = strConnectionString

objConnection.Open

Set objCommand = CreateObject("ADODB.Command")

With objCommand

    .ActiveConnection = objConnection

    .CommandText = strSQL

End With

objCommand.Execute

Set objCommand = Nothing

objConnection.Close

Set objConnection = Nothing
```

4) 步骤、示例 2

1.创建名称为 dbValue 的 WinCC 变量。

2.使用 WINCC_DATA 表和 ID，TagValue 列创建 Access 数据库： ID，创建 TagValue（ID 作用自动值）。

3.设置名称为“SampleDSN”的 ODBC 数据源，引用以上 Access 数据库。

4.编程。

```
Dim objConnection

Dim objCommand

Dim objRecordset

Dim strConnectionString
```

```
Dim strSQL
Dim lngValue
Dim lngCount
strConnectionString = "Provider=MSDASQL;DSN=SampleDSN;UID=;PWD=;"
strSQL = "select TagValue from WINCC_DATA where ID = 1"
Set objConnection = CreateObject("ADODB.Connection")
objConnection.ConnectionString = strConnectionString
objConnection.Open
Set objRecordset = CreateObject("ADODB.Recordset")
Set objCommand = CreateObject("ADODB.Command")
objCommand.ActiveConnection = objConnection
objCommand.CommandText = strSQL
Set objRecordset = objCommand.Execute
lngCount = objRecordset.Fields.Count
If (lngCount>0) Then
objRecordset.movefirst
lngValue = objRecordset.Fields(0).Value
HMIRuntime.Tags("dbValue").Write lngValue
Else
HMIRuntime.Trace "Selection returned no fields" & vbNewLine
End If
Set objCommand = Nothing
objConnection.Close
Set objRecordset = Nothing
Set objConnection = Nothing
```

根据使用的提供程序不同，有多种方法可用来定义连接的连接字符串。

ODBC 的 Microsoft OLE DB 提供程序

启用与 ODBC 数据源的连接。 相应的语法为：

"[Provider=MSDASQL;]{DSN=name|FileDSN=filename};

[DATABASE=database;]UID=user; PWD=password"

其它 Microsoft OLE DB 提供程序（例如 MS Jet、MS SQL Server）可以不使用 DSN 运行。 相应的语法为：

"[Provider=provider;]DRIVER=driver; SERVER=server;

DATABASE=database; UID=user; PWD=password"

5） 实例： 使用 MS 自动化接口

简介

以下三个示例说明了如何使用 MS 自动化接口。

1） 示例 1： MS Excel

本示例中将输入域的输出值写入 Excel 表中。

```
Dim objExcelApp
Set objExcelApp = CreateObject("Excel.Application")
objExcelApp.Visible = True
'ExcelExample.xls is to create before executing this procedure.
'Replace <path> with the real path of the file ExcelExample.xls.
objExcelApp.Workbooks.Open "<path>\ExcelExample.xls"
objExcelApp.Cells(4, 3).Value = ScreenItems("IOField1").OutputValue
objExcelApp.ActiveWorkbook.Save
objExcelApp.Workbooks.Close
objExcelApp.Quit
Set objExcelApp = Nothing
```

2） 示例 2： MS Access

本示例将通过 MS Access 打开报告。

```
Dim objAccessApp
Set objAccessApp = CreateObject("Access.Application")
objAccessApp.Visible = True
'DbSample.mdb and RPT_WINCC_DATA have to create before executing
'this procedure.
'Replace <path> with the real path of the database DbSample.mdb.
objAccessApp.OpenCurrentDatabase "<path>\DbSample.mdb", False
objAccessApp.DoCmd.OpenReport "RPT_WINCC_DATA", 2
objAccessApp.CloseCurrentDatabase
Set objAccessApp = Nothing
```

3) 示例 3： MS Internet Explorer

本示例将打开 MS IE。

```
Dim objIE
Set objIE = CreateObject("InternetExplorer.Application")
objIE.Navigate "http://www.siemens.de"
Do
Loop While objIE.Busy
objIE.Resizable = True
objIE.Width = 500
objIE.Height = 500
objIE.Left = 0
objIE.Top = 0
objIE.Visible = True
```

6) 实例： 启动外部应用程序

简介

以下两个示例说明如何启动外部应用程序。

示例：

```
Dim objWshShell

Set objWshShell = CreateObject("Wscript.Shell")

objWshShell.Run "Notepad Example.txt", 1
```